

**本科实验报告**

|  |  |
| --- | --- |
| 课程名称： | 计算机网络基础 |
| 实验名称： | Lab4 the TCP receiver and the TCP sender |
| 姓 名： |  |
| 学 院： | 计算机学院 |
| 系： |  |
| 专 业： |  |
| 学 号： |  |
| 指导教师： |  |

年 月 日

**浙江大学实验报告**

实验名称： Lab4 the TCP receiver and the TCP sender 实验类型： 设计实验

同组学生： 实验地点：

# 实验目的：

* 学习掌握TCP的工作原理
* 学习掌握流重组器的工作原理
* 学习掌握TCP receiver和TCP sender的相关知识

# 实验内容

* 实现流重组器，一个将字节流的字串或小段按照正确顺序来拼接回连续字节流的模块。
* 实现TCPReceiver。

- 接收TCPsegment；

- 重新组装字节流；

- 确定应该发回发送者的信号，以进行数据确认和流量控制。

* 实现TCPSender：

- 将ByteStream中的数据以TCP报文形式持续发送给接收者；

- 处理TCPReceiver传入的ackno和window size，以追踪接收者当前的接收状态，以及检测丢包的情况；

- 若经过一个超时时间仍然没有接收到TCPReceiver发送的ack包，则重传。

- 发送一个空段

# 主要仪器设备

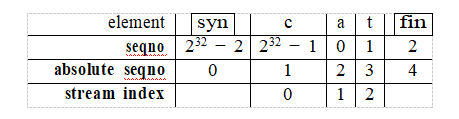
* 联网的PC机
* Linux虚拟机

# 操作方法与实验步骤

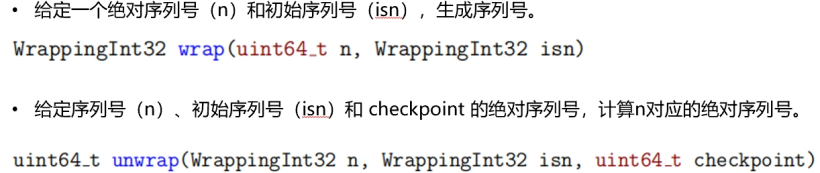
* 理解流重组器StreamReassembler接口法：
* 处理索引问题: 在开始实现receiver和sender前，你需要先处理索引的问题。在流重组器中，每个字节都有一个64位的流索引，流中的第一个字节索引总是0。64位索引足够大，我们可以将其视为不会溢出。然而在TCP报头中，流中的每个字节索引不是用64位索引表示的，而是用32位的序列号表示。这增加了三个复杂性：

1. 你需要考虑到32位的索引的大小问题：通过TCP发送的字节流长度往往没有限制，2^32不是很大。一旦32位的序列号计数到2^32 - 1，流中下一个自己的序列号将为0（容易溢出）。
2. TCP序列号从一个随机值开始：TCP试图确保序列号不会被猜测，也不太可能重复。所以流的序列号不是从零开始的。流中的第一个序列号是一个32位的随机数字，称为初始序列号(ISN)。这是表示SYN(流的开始)的序列号。其余的序列号在此之后增加: 数据的第一个字节将具有ISN+1的序列号(mod 2^32)，第二个字节将具有ISN+2 (mod 2^32)...
3. 逻辑开始和结束各自占用一个序列号: 除了确保接收所有数据字节外，TCP还确保可靠地接收流的开始和结束。因此，在TCP中，SYN(流开始)和FIN(流结束)控制标志也被分配序列号。SYN标志占用的序列号是ISN，流中的每个字节数据也占用一个序列号。SYN和FIN不是流本身的一部分，也不是“字节”—它们代表字节流本身的开始和结束。

综上，有三种序列号，tcp中使用的序列号seqno（32位），流重组器StreamReassembler中使用的序列号stream index（64位），为了二者转换，又引入一种序列号 absolute seqno（64位）。你可以通过下面这张图来理解：



绝对序列号和流索引之间的转换只需加减1即可，但是序列号和绝对序列号之间的转换有点困难，混淆两者可能会产生棘手的错误。为了防止这些错误，我们将用自定义类型**WrappingInt32**表示序列号，并编写它和绝对序列号（用**uint64\_t**表示）之间的转换，实现**wrapping\_intergers.hh**中提供的方法：



**完成代码编写后，在build目录运行ctest -R wrap命令对WrappingInt32进行测试。**

* 实现TCPReceiver

在开始实现receiver的代码前，请复习[TCPSegment](https://cs144.github.io/doc/lab2/class_t_c_p_segment.html)和[TCPHeader](https://cs144.github.io/doc/lab2/struct_t_c_p_header.html)的文档。你可能对[length\_in\_sequence\_space()](https://cs144.github.io/doc/lab2/class_t_c_p_segment.html)方法感兴趣，该方法计算一个段占用多少序列号（包括SYN和FIN标志的序列号）。实现TCPReceiver接口中的方法，具体实现思路参考PPT。

* 实现TCPSender: TCPSender需将序列号、SYN标志、有效负载和FIN标志组成segment发送。然而，TCPSender在收到TCPReceiver发送的ack包时只需读取确认号和窗口大小。实现TCPSender接口中的方法，具体实现思路见PPT。注意，在开始代码实现前，这里有一些关于超时重传的描述，请仔细阅读：
* “时间流逝”：实验中这一概念是通过调用 tick 方法实现的，而不是通过获取实际时间实现的，例如调用tick(5)，说明过去了5ms。
* 超时重传时间 RTO：RTO值会随着网络环境的变化而变化。当TCPSender被构造时，会传入RTO的初始值（RTO保存在\_initial\_retransmission\_timeout）
* 重传计时器：和RTO比较，判断是否超时
* 连续重传计数器：记录连续重传次数，过多的重传次数可能意味着网络的中断

* **实现TCPReceiver和TCPSender后，运行make check\_lab2命令以检查代码的正确性。**
* 温馨提示: 当你在开发代码的时候，可能会遇到无法解决的问题，下面给出解决的办法。

- 运行**cmake .. -DCMAKE\_BUILD\_TYPE=RelASan**命令配置build目录，使编译器能够检测内存错误和未定义的行为并给你很好的诊断。

- 你还可以使用valgrind工具。

- 你也可以运行**cmake .. -DCMAKE\_BUILD\_TYPE=Debug**命令配置并使用GNU调试器（**gdb**）。

- 你可以运行**make clean**和**cmake .. -DCMAKE\_BUILD\_TYPE=Release**命令重置构建系统。

- 如果你不知道如何修复遇到的问你题，你可以运行**rm -rf build**命令删除build目录，创建一个新的build目录并重新运行**cmake..**命令。

# 实验数据记录和处理

以下实验记录均需结合屏幕截图（截取源代码或运行结果），进行文字标注（看完请删除本句）。

* 实现WrappingInt32的关键代码截图
* 测试Wrapping的运行截图
* 实现TCPReceiver的关键代码截图
* 实现TCPSender的关键代码截图
* 运行make check\_lab2命令的测试结果展示

# 实验数据记录和处理

根据你编写的程序运行效果，分别解答以下问题（看完请删除本句）：

* 通过代码，请描述TCPSender是如何发送出一个segment的？

判断是否需要设置SYN位

* 请用自己的理解描述一下TCPSender超时重传的整个流程。
* 请描述一下你为了重传未被确认的段建立的数据结构？为什么？
* 请思考为什么TCPSender实现中有一个发送空段的方法，能描述一下你是怎么理解的吗？

# 实验数据记录和处理